PATENT

METHOD AND SYSTEM FOR INTEGRATING CORES IN FPGA-BASED SYSTEM-ON-CHIP (SoC)

FIELD OF THE INVENTION

[0001] This invention relates generally to programmable logic devices, and more particularly to a method and system for integrating cores for customizing FPGA-based SoCs.

BACKGROUND OF THE INVENTION

[0002] Programmable devices are a class of general-purpose integrated circuits that can be configured for a wide variety of applications. Such programmable devices have two basic versions, mask programmable devices, which are programmed only by a manufacturer, and field programmable devices, which are programmable by the end user. In addition, programmable devices can be further categorized as programmable memory devices or programmable logic devices. Programmable memory devices include programmable read only memory (PROM), erasable programmable read only memory (EPROM) and electronically erasable programmable read only memory (EEPROM). Programmable logic devices include programmable logic array (PLA) devices, programmable array logic (PAL) devices, erasable programmable logic devices (EPLD) devices, and programmable gate arrays (PISA).

[0003] As chip capacity continues to increase significantly, the use of field programmable gate arrays (FPGAs) is quickly replacing the use of application specific integrated circuits (ASICs). An ASIC is a specialized integrated circuit that is designed for a particular application and can be implemented as a specialized microprocessor. Notably, a FPGA is a programmable logic device (PLD) that has an extremely high density of electronic gates as compared to an ASIC. This

high gate density has contributed immensely to the popularity of FPGAs. Notably, FPGAs can be designed using a variety of architectures that can include user configurable input/output blocks (IOBs), and programmable logic blocks having configurable interconnects and switching capability.

[0004] The advancement of computer chip technology has also resulted in the development of embedded processors and controllers. An embedded processor or controller can be a microprocessor or microcontroller circuitry that has been integrated into an electronic device as opposed to being built as a standalone module or "plugin card." Advancement of FPGA technology has led to the development of FPGA-based system-on-chips (SoC) including FPGA-based embedded processor SoCs. A SoC is a fully functional product having its electronic circuitry contained on a single chip. While a microprocessor chip requires ancillary hardware electronic components to process instructions, a SoC would include all required ancillary electronics. For example, a SoC for a cellular telephone can include a microprocessor, encoder, decoder, digital signal processor (DSP), RAM and ROM. It should be understood within contemplation of the present invention that an FPGA-Based SoC does not necessarily include a microprocessor or microcontroller. For example, a SoC for a cellular telephone could also include an encoder, decoder, digital signal processor (DSP), RAM and ROM that rely on an external microprocessor. It should also be understood herein that "FPGA-based embedded processor SoCs" are a specific subset of FPGA-based SoCs that would include their own processors.

[0005] In order for device manufacturers to develop FPGA-based SoCs or FPGA-based embedded processor SoCs, it is necessary for them to acquire intellectual property rights for system components and/or related technologies that are utilized to create the FPGA-based SoCs. These system

components and/or technologies are called cores or Intellectual Property (IP) cores. An electronic file containing system component information can typically be used to represent the core. A device manufacturer will generally acquire several cores that are integrated to fabricate the SoC.

[0006] Notwithstanding advantages provided by using FPGA-based SoCs, the development of these SoCs can be very challenging. Although a vast proportion of cores are commercially available, a significantly greater proportion of cores are proprietary. Proprietary cores can be called customer specific cores. Commercially available cores can typically include standardized interfaces, which can provide interconnectivity between system components from various vendors. Customer specific cores can typically include proprietary interfaces that do not readily facilitate interconnectivity between system components from other vendors. For example, customer specific cores can be written in proprietary languages, which are completely different from standardized languages. Since customer specific cores do not readily facilitate interconnectivity with other vendor's system components, integrating customer specific cores during customization of an FPGA-based SoC can be time consuming. This resulted in increased development cost and greater time-to-market. Integration of the cores can include simulating, modeling and debugging the integrated cores in an operating environment. Simulation and modeling can be a daunting task since it can take hours if not days to simulate a few milliseconds of real time operation. FPGA based embedded processor SoCs are being introduced into the market, but there are no solutions which allow users to customize the system, the hardware cores, and the associated software nor is there a system enabling a user to tradeoff a function which is implemented in hardware (FPGA fabric) or software

(running on the embedded processor). It would be desirable to have a method and system for better integrating cores during customization of FPGA-based SoCs and that further overcome the shortcomings described above.

SUMMARY OF THE INVENTION

[0007] The invention can provide a method for integrating system component cores during customization of a FPGA-based SoC. Subsequent to selecting a system component used for customizing the FPGA-based SoC, parameters can be used to configure the selected system component for use with the FPGA-based SoC. The parameters used to configure the selected system component can be propagated and used to configure peer system components. Notably, other parameters that are used to configure the peer system component can also be propagated and used to configure the selected system component. The parameters used to configure the peer system components can be propagated to subsequently selected system components that can be used to configure the FPGA-based SoC. Selection of the system components can also include the provision of an option for selecting a hardware implementation or a software implementation for customizing the FPGA-based SoC. Additionally, the step of selecting the system component can include selecting a system component from the group consisting of a hardware core and a software core. The invention can also provide an interface for integrating hardware or software system component cores used for customizing an FPGA-based SoC.

[0008] In a further aspect of the invention, an interface is provided for integrating hardware system component cores used for customizing an FPGA-based SoC. The interface can include a slave connection circuitry communicatively interfaced to a processor bus and a master connection circuitry communicatively interfaced to the processor bus. A

multiplexer (MUX) can facilitate selecting the slave connection circuitry or the master connection circuitry in order to provide communication between the processor bus and a selected hardware system component core used for customizing the FPGA-based SoC. The selected hardware system component core can be either a proprietary customer specific hardware core or a commercially available hardware core.

[0009] In accordance with the inventive arrangements, the interface can further include a direct memory access (DMA) controller for providing direct access to a memory device. A write buffer and a read buffer can provide temporary storage of I/O data, which can be communicated between the memory device and the selected hardware system component core. An interrupt controller coupled to a multiplexer can latch individual interrupt signals and provide a single signal to a microprocessor indicating an interrupt condition. The microprocessor then typically clears the interrupt condition that is latched in the interrupt controller after servicing the interrupt in a peripheral.

[0010] In yet a further aspect of the invention, a GUI for integrating system component cores during customization of a FPGA-based SoC is provided. The GUI can include a selection object for selecting a system component core for customizing the FPGA-based SoC. A configuration object can facilitate configuration of selected system component core and peer system components. A display object can be used to facilitate display of parameters used by the configuration object to configure selected system components and peer system components. The configuration object can further include a parameter distribution object for distributing the parameters to subsequently selected system component cores and peer system components. The selection object can further include a dialog for selecting hardware system component cores and software system component cores.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] FIG. 1 is a block diagram of a processor system generator in accordance with the invention.

[0012] FIG. 2 depicts an exemplary topological view of the system model in accordance with the inventive arrangements.

[0013] FIG. 3 depicts a flow chart illustrating exemplary steps for integrating system component cores in accordance with the invention.

[0014] FIG. 4 depicts an interface for integrating software system component cores in accordance with the inventive arrangements.

[0015] FIG. 5 depicts an exemplary hardware interface for integrating hardware system components in accordance with the invention.

[0016] FIG. 6 depicts and exemplary GUI in accordance with the present invention.

DETAILED DESCRIPTION OF THE DRAWINGS

[0017] Referring to FIG. 1, there is shown a block diagram illustrating an exemplary system for developing and verifying a FPGA-based SoC in accordance with the invention. For illustrative purposes, and without limiting the scope of the invention, an embedded system consisting of a microprocessor, buses, memory architecture, peripherals, and software components is presented, although a system using an external microprocessor is certainly contemplated within the scope of the invention. Exemplary software components for the embedded system can include, but is not limited to, device drivers and system software, such as a real time operating system (RTOS) and protocol stacks. An exemplary development environment for this embedded system can include, but is not limited to, one or more libraries for microprocessors,

peripherals, system software, and device drivers. The relevant bus architectures and memory options that can be utilized for the development of an FPGA-based SoC can be included in the libraries. Further, a good example of hardware/software function tradeoff can involve the protocol stack, which can be implemented in either hardware or software. A user may implement a protocol stack in software if there is sufficient processing power to meet all performance requirements or the user could implement the protocol stack in hardware given sufficient FPGA resources and a presumed need for higher performance.

[0018] In general, a system model can be created to facilitate design and testing of an FPGA-based SoC. The system model can include a data structure that represents the internal structure and functionality of the FPGA-based SoC. The system model can include, but is not limited to, system components, interconnections between components, and attributes, which define various characteristics and functionality of the system components and interconnections. The data structure can be a hierarchical structure, for example a tree structure, which can mirror the design hierarchy of the embedded system. This system model can also include algorithms, which can facilitate selection and customization of system components. Notably, the system model can be part of an integrated object-oriented system (OOS) that can facilitate selection and customization of the system components. Alternatively, other mechanisms and algorithms external to the system model can facilitate selection and customization of the system components.

[0019] Referring now to FIG. 1, there are shown a platform generator 105, a system selector 110, a system customizer 115, a system analyzer 120, a code generator 125 and a system implementor 130 all forming a processor system generator. The platform generator 105 can include one or more GUIs that

can facilitate design of the system model. A main GUI can provide various system options and informational dialogs. The platform generator can include, a navigator GUI having one or more dialogs and/or objects, a topological GUI having one or more dialogs and/or objects and a selection customizer GUI having one or more dialogs and/or objects. One or more dialogs and/or objects can provide system component resource counts, performance estimates, power requirements and system and application software requirements. For example, a GUI can be used to display a table or chart representing the resource allocation for the system components. Advantageously, such table or chart can provide an easily readable condensed view of the system resource allocation. An exemplary table is illustrated below.

Device	**LUTs**	**DFFs**	**Slices**	**BRAM**	**I/Os**
OPB Arbiter	300	200	200	0	9
UART 16450	500	400	300	0	12
Ethernet 10/100M	2500	1700	1500	0	12
Total Utilized	3300	2300	2000	0	21
Device Resources	122880	122880	61440	3456	1488
Available Resources	119580	120580	59440	3456	1267

[0020] Referring to the table, a condensed view of the system resources is provided. Specifically, the table shows a breakdown of particular resources utilized by each device and also the total resources utilized by all devices. The available resources can be computed based on the total utilized resources and the total device resources. For example, there are 122880 D-flip flops (D-FFs) available.

OPB arbiter utilizes 200 D-FFs, UART 16450 utilizes 400 D-FFs, and Ethernet 10/100M device utilizes 1700 D-FFs. Hence, there are 2300 D-FFs utilized, which leaves 120580 available.

[0021] The navigator dialog and/or object can provide an interactive interface that can facilitate viewing of design specification and configuration information. For example, one or more navigator objects can provide a graphical view to facilitate the insertion of a microprocessor from a library into the system model. In a case where a universal asynchronous receiver/transmitter (UART) is selected as a peripheral, the navigator object and/or dialog can permit customization of the UART. The navigator dialog can also be configured to permit switching between multiple design and implementation tasks. The topological dialog can utilize a block diagram format to provide a topological view that can visually represent the existing state of the system model. The selection customizer object can permit the selection and customization of a system component. Upon selection of a system component, a GUI which can include a dialog and can facilitate customization of the system component. Platform generator 105 can have the capability to permit a particular state and/or stage of the system design and implementation to be saved and recalled at a subsequent time.

[0022] System selector 110 can be a GUI that can facilitate selection of the system components that can be used to design the FPGA-based SoC. For example, the system selector 110 can provide one or more dialogs that can permit the selection of microprocessors, microcontrollers, peripheral devices, buses, system software and application software. During selection of system components, each of the selected components can be independently treated.

[0023] The system customizer 115 can include one or more GUIs having objects and/or dialogs that can facilitate customization or configuration of system components and

software. Referring to FIG. 1, there are shown a system parameter customizer 115a, a hardware intellectual property (IP) parameter customizer 115b, and a software IP parameter customizer 115c. The system parameter customizer 115a can facilitate customization of the memory map, interrupt bindings and priorities, and global and default system parameter definitions. The hardware intellectual property (IP) parameter customizer 115b can facilitate customization of device specific parameters. For example, data bus widths, IP interfaces and device specific parameters can be customized by hardware intellectual property (IP) parameter customizer 115b.

[0024] The software intellectual property (IP) parameter customizer 115c can facilitate customization of software specific parameters. For example, upon selection of a system component or a peripheral, an interrupt request (IRQ) number, a memory mapped I/O address and default initialization parameters can be assigned to the peripheral by the software IP parameter customizer 115c. In a case where a UART has been selected as a peripheral, default parameters can include, but are not limited to, stop bits, parity designation on/off, and baud rate. The customizer system 115 not only provides selection of the system components, but can also be configured to bind system parameters to system components. For example, the memory map for a particular peripheral can be bound to the peripheral giving the peripheral its unique memory address space. Furthermore, a GUI having one or more dialogs can be used to populate a system model data structure with customization parameters and/or attributes.

[0025] The system analyzer 120 can include one or more GUIs having objects and/or dialogs that can provide immediate feedback regarding architectural choices made during customization. The system analyzer 120 can include software

that can have the capability to validate and analyze the system model while it is being customized. If problems including, incompatibilities, conflicts and/or system violations occur, the system analyzer 120 can issue immediate warnings and/or provide possible solutions. The system analyzer 120 can perform tasks such as system checks, parameter consistency checks, data type and value propagation checks, interconnection inference, and resource and performance analysis. Interconnection reference pertains to implications that can result from making certain connections. The system analyzer 120 can also assign device identifications (IDs) to system components and computing configuration read-only-memory (ROM) data. Exemplary system and parameter consistency checks can include, matching data bus widths of peripherals and system components, determining interrupt conflicts, determining memory map conflicts, determining memory size and usage, determining device counts, determining availability of FPGA resources and determining maximum operating frequency.

[0026] The system analyzer 120 can be configured to propagate default values, global values and/or previously defined values through the system model. For example, if a bus is configured with a default data width of 16 bits, then each peripheral that "sits on" or utilizes that bus can automatically be configured with a data width of 16 bits. It should be recognized by one skilled in the art that although a peripheral device may be automatically configured with the default bus width value, this value can be overwritten. For example, depending on the application, availability of certain devices can dictate that two (2) 8-bit devices be utilized instead of a single 16-bit device. Advantageously, the propagation of values can prevent multiple entries of similar data which typically increases development time.